МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ
ФАКУЛЬТЕТ КІБЕРБЕЗПЕКИ, КОМП'ЮТЕРНОЇ ТА ПРОГРАМНОЇ ІНЖЕНЕРІЇ
КАФЕДРА ПРИКЛАДНОЇ ІНФОРМАТИКИ

ДОПУСТИТИ ДО ЗАХИСТУ

Завідувач кафедри

_____ __Гамаюн В.П.
(підпис) (ПІБ)

"_____"_____2021р.

ДИПЛОМНИЙ ПРОЕКТ
(ПОЯСНЮВАЛЬНА ЗАПИСКА)

ВИПУСКНИКА ОСВІТНЬОГО СТУПЕНЯ "БАКАЛАВР"

Тема: Андроїд-додаток для сервісу таксі_____

Виконавець: _____ Сапіжак Едуард Васильович
(підпис) **(ПІБ)**

Керівник: _____ Водоп'янов Сергій В'ячеславович
(підпис) **(ПІБ)**

Нормоконтролер: _____ Боровик Володимир Миколайович
(підпис) **(ПІБ)**

Київ 2021

ВСТУП

В сучасних умовах важко представити життя людини без смартфону або будь-якого іншого портативного пристрою. Раніше телефони виконували лише роль засобу для зв'язку з іншими людьми. Зараз мобільний телефон – це універсальний інструмент, який виконує широкий спектр обчислювальних задач. Він поміщається в кишеню та має дуже багато корисних функцій, які значно спрощують життя людини. З розвитком мобільних пристроїв зростає і об'єм програмного забезпечення під мобільні платформи, тому сучасні компанії все більше приділяють увагу розробці мобільних додатків, які дозволяють оптимізувати роботу бізнесу. Це стосується і компаній, які займаються таксомоторними перевезеннями. В сучасному світі жителі великих міст зі жвавим дорожнім трафіком все більше віддають перевагу користуванню послугами таксі, тому що пересування громадським транспортом не є комфортним, а придбання і подальше технічне обслуговування власного автомобіля потребує немалих коштів.

Традиційні способи бронювання авто, в яких потрібно зв'язуватись з оператором таксомоторної компанії або ходити по місту та шукати таксі, швидко втрачають свою актуальність з розвитком ІТ-індустрії. Споживачі більше віддають перевагу сервісам, які працюють з мобільними додатками. Головна причина такої уваги – це зручність. Додатки для виклику таксі дозволяють повністю автоматизувати процес замовлення авто, що зменшує витрати як зі сторони сервісу, так і зі сторони клієнта, а саме:

- автоматизація дозволяє значно скоротити штат працівників компанії;
- не потрібно витрачати час на пошук клієнта або водія;
- в додатку за допомогою картографічних ресурсів обирається пункт призначення і одразу відображається вартість поїздки, яка залежить від дальності маршруту та різних послуг;
- оплатити поїздку можна одразу в додатку за допомогою банківської карти;
- присутня система профілів та рейтингу. Це дозволяє відсіювати поганих водіїв або пасажирів.

РОЗДІЛ 1
АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1. Дослідження методів розробки мобільного додатку для сервісу таксі

Розробка мобільного додатку для сервісу таксі включає в себе такі етапи:

1) *Аналіз.* На цьому етапі детально вивчається ринок схожого програмного забезпечення, визначається цільова аудиторія, функціонал, який необхідно реалізувати в додатку, проводяться розрахунки. Після цього формується загальна концепція додатку і розробляється план робіт.

2) *Проектування.* На основі зібраних на першому етапі даних команда починає розробку програмного забезпечення. Головне завдання – сформулювати задачі та цілі, розробити структуру, відібрати технології та методи для реалізації проекту.

3) *Дизайн*. На цьому етапі дизайнером розробляється важлива частина додатку – його інтерфейс. Він повинен бути простим, приємним та інтуїтивно зрозумілим для звичайного користувача і коректно відображатися на різних мобільних пристроях.

4) *Розробка*. Після виготовлення дизайнером прототипу, розробники беруться за написання коду. Це може бути розробка під конкретну операційну систему або створення кроссплатформенного додатку. На цьому етапі реалізується весь функціонал, який обговорювався на етапі аналізу, і після цього проводиться тестування. Коли додаток пройде тестування, можна переходити до його запуску на різних платформах.

На самому початку треба визначити, для кого створюється додаток. Перший варіант – служба таксі, яка вже працює і займає певне місце на ринку. В такому випадку при розробці потрібно враховувати налагодженні процеси, що проходять в компанії. Другий варіант – стартап-проект. Це означає, що додаток створюється для нової компанії, яка почне свою діяльність одразу зі своїм власним додатком.

Наступним етапом обираємо масштаб, який буде охоплювати наш сервіс: поїздки будуть лише в межах міста чи всієї країни.

Розробка додатку повинна починатися з опису всіх сценаріїв взаємодії користувача і сервісу. Після цього формується чітке розуміння, що потрібен автоматизувати додаток. На основі цього створюється програмний інтерфейс додатку (API).

Необхідно реалізувати систему реєстрації користувачів. Це можуть бути як і пасажири, так і водії. Для спрощення цього процесу додають можливість реєстрації за допомогою соціальних мереж, таких як Google+, Facebook тощо.

Основою додатку є інтеграція з картографічними ресурсами. Карти та GPS використовуються для визначення теперішнього місцезнаходження водія або клієнта і побудови маршруту. Для зменшення затрат ресурсів слід використовувати вбудовані карти платформи, для якої розробляється клієнт (наприклад, на Android це буде Google Maps).

Для зручності треба реалізувати можливість оплати послуг всередині додатку (наприклад, через дані банківської карти або Google Pay).

Важливою частиною є система рейтингу водіїв. Вона допомагає відслідковувати рівень задоволення клієнтів і виявляти проблеми, пов’язані з якістю обслуговування.

Ще один фактор – вибір платформи. На початку слід створити додаток для більш поширеної платформи. За даними статистики веб-сайту StatCounter Global Stats на квітень 2021 року ринок мобільних пристроїв на різних операційних системах виглядає так:

- США: IOS – 59,13%, Android – 40,58%, Samsung – 0,21%, Windows – 0,03%;

- Великобританія: IOS – 5,44%, Android – 49%, Samsung – 0,5%, Windows – 0,03%;
- Індія: Android – 95,77%, IOS – 2,97%, KaiOS – 0,99%, Samsung – 0,13%;
- Африка: Android – 79,29%, IOS – 18,62%, Unknown – 1,43%, Samsung – 0,24%;
- Європа: Android – 68,82%, IOS – 30,39%, Samsung – 0,66%, Windows – 0,06%;
- Україна: Android – 80,49%, IOS – 19,16%, Samsung – 0,17%, Windows – 0,07%.

Частки операційних систем в світовому масштабі за даними статистики веб-сайту StatCounter Global Stats зображені на рис. 1.1.

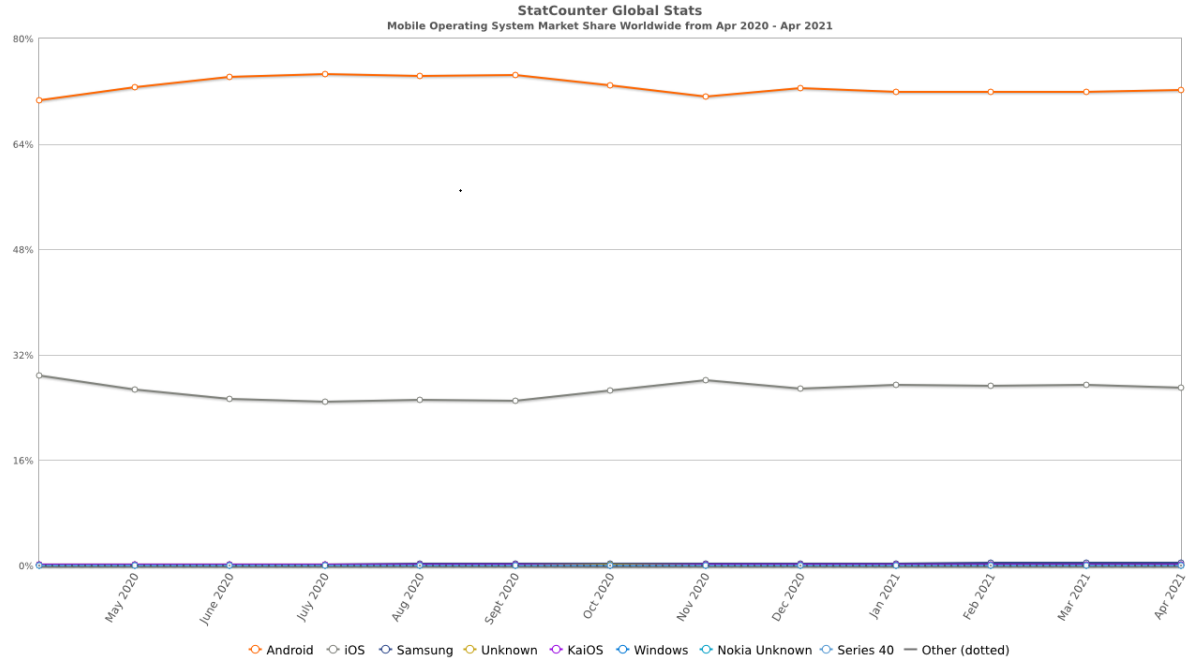


Рис. 1.1. Графік поширення операційних систем на мобільних пристроях в світовому масштабі

На конференції розробників «Google I/O 2019» представники компанії Google заявили, що в кількісних показниках на ринку існує більше 2,5 мільярдів активних Android-пристроїв. Це робить Android самою поширеною операційною системою по

кількості користувачів. Тому вибір Android платформи для розробки додатку можна вважати виправданим.

При розробці мобільного додатку для виклику таксі рекомендується використовувати наступні програмні засоби та платформи:

- Google Maps API (геолокація);
- Firebase (база даних та служба автентифікації);

Для реалізації відображення карти в мобільному додатку, потрібна інтеграція з Google Maps за допомогою ключа API. Цей ключ потрібен для підключення проекту до сервісів Google.

Firebase – це платформа з великим набором функцій, які можна використовуватися при розробці власного мобільного додатку. Вона пришвидшує розробку прихованої від користувача програмної частини проекту. Це оптимізує процес створення мобільних додатків та дає можливість повністю зосередитись на інтерфейсі користувача.

Firebase надає багато можливостей розробникам програмного забезпечення. В цю платформу входить сервер, БД, хостинг, автентифікація.

Firebase Realtime Database надає API, який синхронізує дані додатку між клієнтами та зберігає їх в хмарі. Додаток підключається до хмарної БД через протокол WebSocket, який відповідає за синхронізацію даних .

Firebase Realtime Database – це база даних NoSQL, тому вона відрізняється від реляційних баз даних за функціональністю. Програмний інтерфейс БД в реальному часі дозволяє виконувати лише ті операції, які можуть виконатися швидкою. Така можливість дозволяє обробляти дані в реальному часі. Але важливо визначити, яким чином користувачі будуть отримувати доступ до даних та розробити структуру.

Також Firebase надає можливість зберігання файлів в хмарному сховищі, яке забезпечує надійне завантаження файлів клієнтом або сервером. Всі файли захищені надійною системою безпеки.

Служба автентифікації Firebase Auth надає можливість автентифікації користувача в мобільному додатку за допомогою електронної пошти та паролю.

Підтримує відкритий протокол авторизації OAuth 2.0. Система автентифікації реалізується за допомогою хмарної бази даних.

Використання платформи Firebase дозволяє зменшити час, який витрачається на розробку, та підвищити ефективність роботи мобільного додатку. Особливості платформи:

- збільшує швидкість роботи мобільного додатку;
- зменшує в багато разів ймовірність збоїв;
- дозволяє переглядати статистику використання додатку та підтримує зворотній зв'язок;
- ріст кількості користувачів не потребує втручання в код серверу.

В додатку не повинно слабких місць. Зміст та дизайн – це найважливіші частини програми, які впливають на сприймання користувачем додатку. Тому на етапі проектування потрібно приділити увагу формуванню вимог до інтерфейсу користувача.

Ключові аспекти створення додатків для сервісу таксі:

- складний функціонал. Для того, щоб додаток працював стабільно і витримував мережеві навантаження, потрібно створити клієнт-серверну архітектуру, що значно понизить вимоги до ЕОМ;
- для оптимізації можна вилучити адміністративну частину і реалізувати можливість зв'язку між водієм та пасажиром в самому додатку.

1.2. Огляд схожих застосунків

1.2.1. Uber

Uber – це міжнародна компанія, яка на сьогоднішній день активно працює в Україні. Uber був заснований Гаретом Грампом та Тревісом Каланіком в 2009 році. Компанія Uber має два окремих додатка для водіїв так клієнтів. Додаток для клієнтів зручний та простий у використанні. Є можливість обирати клас автомобіля, відслідковувати його на карті та обирати час, на який потрібно подати таксі. В додатку можна легко вказати особливості поїздки (наприклад, з багажем). Клієнт

також може замовити кур’єрську доставку. Головний екран мобільного додатку Uber для пассажирів зображений на рис. 1.2..

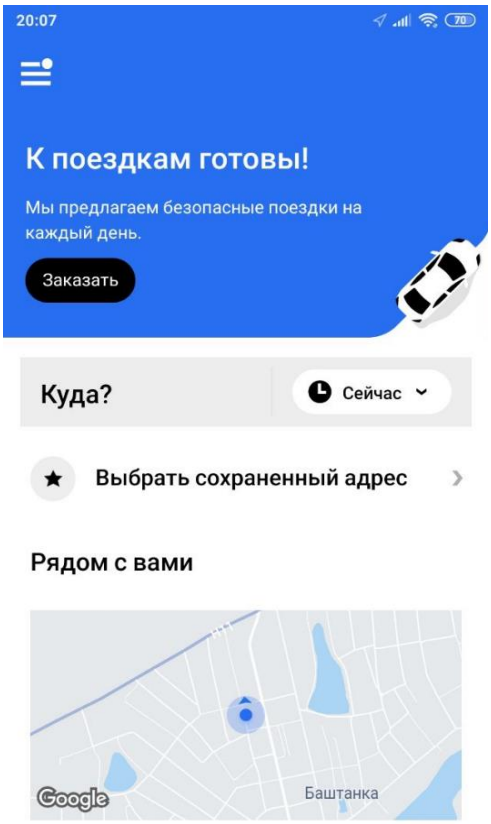


Рис. 1.2. Головний екран мобільного додатку Uber

Додаток Uber працює наступним чином:

- клієнт відкриває додаток вводить пункт призначення на карті, обирає тип поїздки з урахування різних послуг, а потім підтверджує своє замовлення;
- водій, що знаходиться неподалік, отримує замовлення. Клієнт отримує повідомлення коли прибуває таксі;
- за допомогою навігатору, який інтегрований в додаток, водій може легко орієнтуватися при плануванні маршруту;
- після поїздки в додатку водій так клієнт можуть поставити оцінку один одному.

1.2.2. Uklon

Uklon – одна з перших компаній, яка впровадила додаток для виклику таксі на ринку України. Вона має два окремих додатка для водіїв так клієнтів. Особливістю додатку для клієнтів є можливість спілкування з водієм через чат, вказувати декілька маршрутів та повторювати попередні. Для того щоб зацікавити водія, клієнт може збільшити вартість поїздки. Головний екран мобільного додатку Uklon для клієнтів зображений на рис. 1.3..

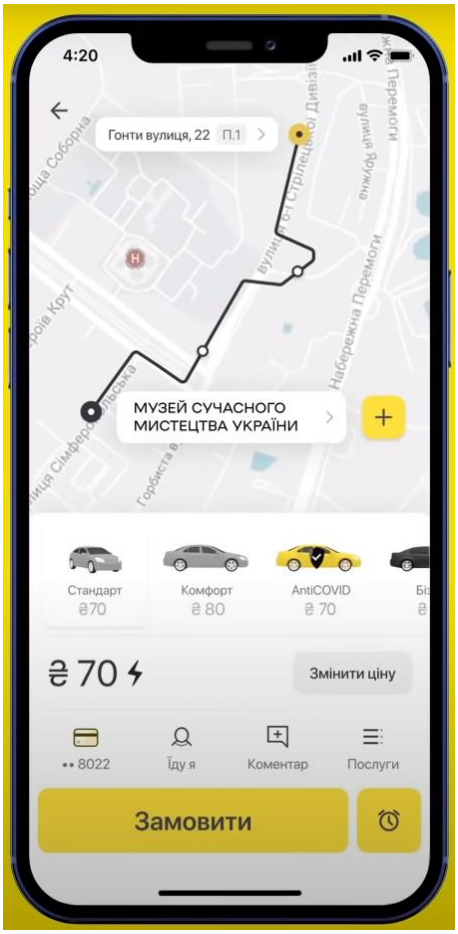


Рис. 1.3. Головний екран мобільного додатку Uklon

Особливості сервісу Uklon:

- швидкий виклик авто – сервіс підбирає авто по вказаним параметрам;
- послуга доставки товарів;
- пошук місць поруч з вами;
- створення попереднього замовлення;

- історія замовлень;
- збереження адресу .

1.2.3. Таксі Експрес

Через додаток Таксі Експрес можливо замовити таксі, обрати клас авто та обрати час для подачі машини. Є можливість найняти водія для поїздки на особистому транспорті. Клієнт може також замовити вантажне таксі або таксі без розпізнавальних знаків. Головний екран мобільного додатку Таксі Експрес зображений на рис. 1.4..

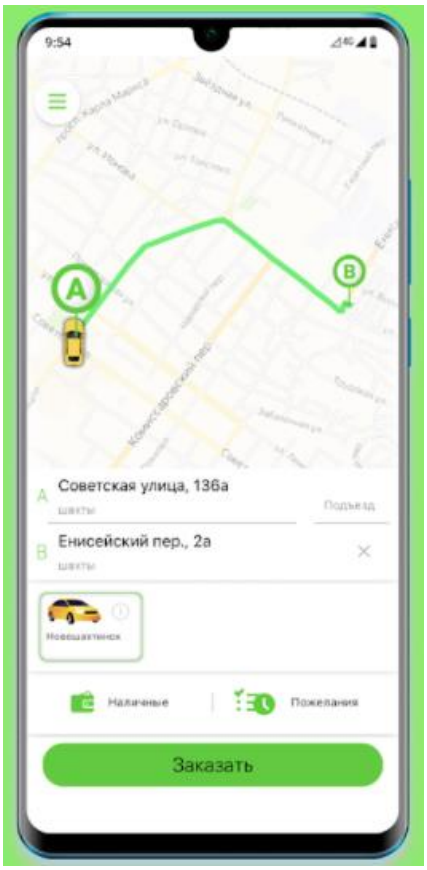


Рис. 1.4. Інтерфейс додатку Таксі Експрес

1.2.4. Bolt

Bolt – естонська приватна компанія, яка надає послуги пошуку й замовлення таксі. Компанія була заснована Маркусом Віллігом в 2013 році під назвою «Taxify» з метою об’єднати всі служби таксі на одній платформі. В Україні сервіс був

запущений повноцінно в 2018 році. Існує два застосунки Bolt: для пасажирів та водіїв. На рис. 1.5. зображений інтерфейс додатку Bolt для пасажирів.

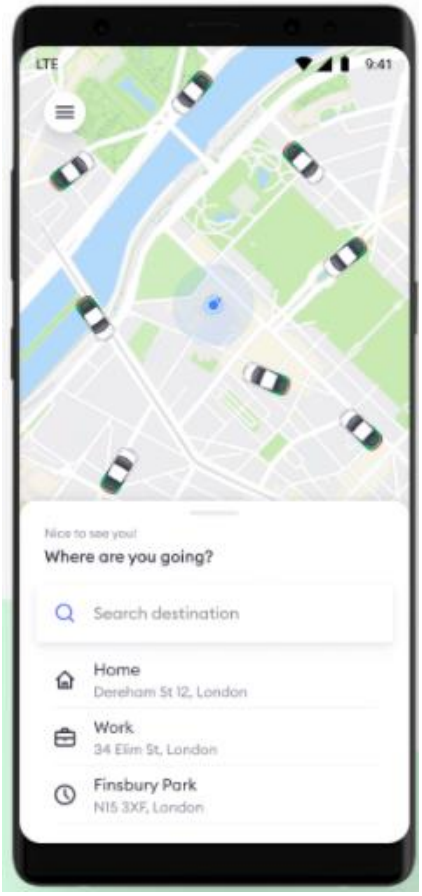


Рис. 1.5. Інтерфейс додатку Bolt для пасажирів

Висновки до розділу

Провівши дослідження методів створення мобільного додатку для сервісу таксі, було визначено основі етапи розробки програмного забезпечення: аналіз ринку подібних застосунків, проектування додатку, розробка дизайну інтерфейсу, реалізація додатку.

В ході аналізу подібних мобільних застосунків було визначено основний функціонал, який повинен бути реалізований в додатку. Був зроблений висновок, що всі компанії, які створювали мобільні додатки для таксомоторних перевезень, мають два окремих застосунка: для водіїв так клієнтів. Тому, розробка буде направлена на створення одного додатку, яким зможуть користуватися як водії, так і клієнти.

РОЗДІЛ 2
ПРОЕКТУВАННЯ МОБІЛЬНОГО ДОДАТКУ

На етапі проектування системи автоматизованого замовлення таксі визначаються функціональні та нефункціональні вимоги, вимоги до інтерфейсу користувача мобільного додатку. На основі зібраних вимог створюється діаграма варіантів використання системи, структура мобільного додатку, проектується архітектура програми та база даних.

2.1. Функціональні вимоги

Система для замовлення таксі повинна задовольняти наступним вимогам:

- система повинна бути призначена для водіїв та клієнтів;
- система повинна надавати користувачу можливість реєстрації та авторизації за допомогою електронної пошти і паролю;
- система повинна забезпечувати синхронізацію даних різних клієнтів в реальному часі;
- система повинна надавати користувачу можливість змінювати свої персональні дані;
- система повинна надавати можливість клієнту замовити таксі;
- система повинна демонструвати переміщення водія та клієнта на карті;
- система повинна надавати клієнту інформацію про водія;
- система повинна надавати водію інформацію про клієнта;
- система повинна надавати можливість телефонного зв’язку водія з клієнтом;
- система повинна надавати можливість клієнту відмовитись від поїздки;
- система повинна забезпечувати збереження сесії користувача;

- система повинна надавати можливість користувачу виходити з облікового запису.

2.2. Нефункціональні вимоги

В ході аналізу функціональних вимог та огляду аналогів даного проекту були сформовані наступні нефункціональні вимоги:

- мобільний додаток повинен бути написаний на мові програмування Java;
- мобільний додаток повинен бути написаний під операційну систему Android версії 6.0 та вище;
- сервер та база даних мобільного додатку повинні бути розміщенні на платформі Firebase.

2.3. Вимоги до інтерфейсу користувача

Інтерфейс користувача мобільного додатку повинен відповідати таким вимогам:

- зручне представлення даних;
- можливість індивідуальних налаштувань для окремого користувача;
- використання звичних елементів інтерфейсу користувача: прямокутні кнопки, звичне розташування меню;
- важливі елементи повинні знаходитися на першому плані;
- повинен бути заклик до дії, де вона необхідна.

Для підвищення якості та ефективності інтерфейсу користувача мобільний додаток повинен відповідати наступним критеріям:

1) *Інтуїтивно-зрозумілий*. При першому вході в додаток користувач повинен легко і самостійно розібратися з функціоналом додатку.

2) *Швидкий доступ*. Важливим є те, скільки часу витрачає користувач на пошук необхідної інформації або функції. Тому необхідно провести оптимізацію інтерфейсу і зробити його більш простим та зрозумілим.

3) *Емоції.* Проводячи час в додатку, користувач повинен отримувати задоволення від цього процесу. Щоб зробити інтерфейс більш дружнім, потрібно гармонічно використовувати кольори та форми об’єктів.

При створенні інтерфейсу користувача мобільного додатку треба дотримуватись таких правил:

1) *Думати як користувач.* Треба визначити, хто буде користуватися додатком і з якою метою він це буде робити. Якщо це додаток для продажу товарів, головною кнопкою буде “Купити”. Якщо це додаток з новинами, то зручна навігація – головний елемент. Важливо розуміти цільову аудиторію свого застосунку.

2) *Нічого зайвого.* Для того, щоб користувач зміг орієнтуватися в додатку, інтерфейс повинен бути легким та зрозумілим. Нічного зайвого, лише важлива частина.

3) *Всі взаємопов’язані елементи повинні бути логічно з’єднані.* Основою комфортної взаємодії користувача з додатком є логіка. Всі елементи повинні бути з’єднані і логічно зв’язані.

4) *Контекст використання є теж важливим.* Потрібно враховувати при яких умовах будуть користуватися вашим додатком. Наприклад, таксі найчастіше замовляють на вулиці.

5) *Ієрархія по важливості.* Важливі елементи повинні розташовуватись в лівій частині додатку. Це пов’язано з тим, що більшість людей читає зліва направо. В правому нижньому кутку – найменш важливі елементи. Важливо дотримуватися ієрархії розташування елементів, тоді програма буду органічною та зручною у використанні.

При розробці мобільного додатку під платформу Android інтерфейс користувача повинен відповідати концепту створення візуальних та рухомих елементів під назвою «Material Design».

«Material Design» – дизайн програмного забезпечення і додатків операційної системи Android від компанії Google. Вперше був представлений на конференції «Google I/O» 25 липня 2014 року. Ідея дизайну полягає в додатках, які

відкриваються і згортаються як паперові картки, використовуючи ефект тіней (рис. 2.1.).

Рис. 2.1. Порівняння «Material Design» та звичайного дизайну

У додатку не повинно бути гострих кутів, картки повинні змінюватися плавно. Таке відображення інтерфейсу (рис. 2.2.) корисне тим, що оптимізує роботу зі сторінками, значно спрощує масштабування інтерфейсу під різні екрани пристроїв і дозволяє швидко сприймати потрібний вид контенту в потоці.

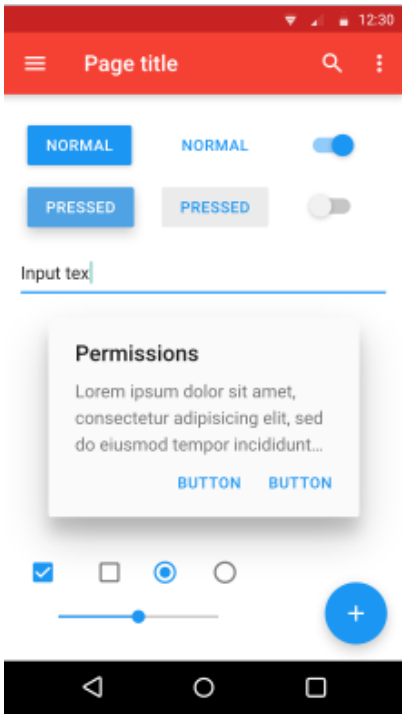


Рис. 2.2. Основні компоненти дизайну «Material Design»

«Material Design» будується на трьох принципах:

1) *Матеріал є метафора.* Метафора матеріалу – це об'єднуюча теорія, яка раціоналізує простір і систему руху. Світло, поверхні та рух – ключові елементи для відображення руху, вони взаємодіють між собою та існують в просторі відносного один одного. Завдяки освітленню формуються тіні, які ділять простір та виділяють рухомі частини.

2) *Поліграфічний дизайн.* Основі елементи дизайну – колір, масштаб, сітки, простір, типографія. Вони створюють фокус, значення та ієрархію. Робиться акцент на основних функціональних можливостях, які одразу попадають в поле зору користувача.

3) *Дія передбачає сенс.* Початкові дії користувача ініціюють рух, який змінює всю конструкцію.

2.4. Варіанти використання системи

Діаграма варіантів використання мобільного додатку представлена на рис 2.3.. Система автоматизованого замовлення таксі містить два актора, які взаємодіють із системою та між собою за допомогою мобільного телефону. Перший актор – це клієнт, а другий – водій.

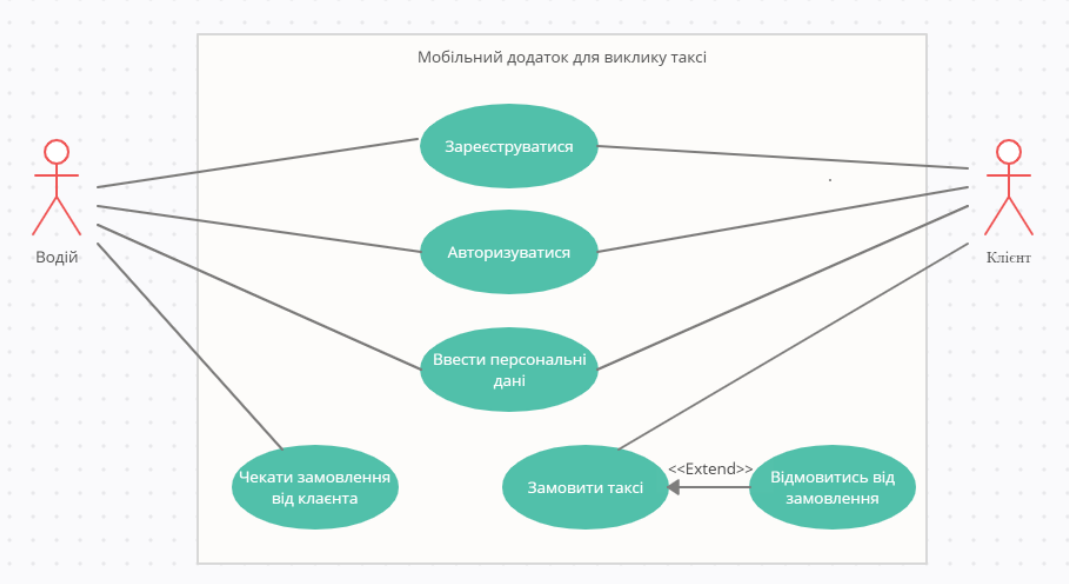


Рис. 2.3. Варіанти використання мобільного додатку для замовлення таксі

Водій може *зареєструватися* в додатку, вказавши при цьому електронну пошту та пароль.

Водій може *авторизуватися* в додатку, вказавши при цьому електронну пошту та пароль.

Водій може *ввести персональні дані* після авторизації в додатку.

Водій може *чекати замовлення від клієнта* після авторизації в додатку.

Клієнт може *зареєструватися* в додатку, вказавши при цьому електронну пошту та пароль.

Клієнт може *авторизуватися* в додатку, вказавши при цьому електронну пошту та пароль.

Клієнт може *ввести персональні дані* після авторизації в додатку.

Клієнт може *замовити таксі* після авторизації в додатку.

Клієнт може *відмовитись від замовлення* таксі.

2.5. Структура мобільного додатку

Мобільного додаток для виклику таксі складається з семи екранів:

- стартовий екран;
- екран авторизації;
- екран авторизації для водія;
- екран авторизації для клієнта ;
- екран водія;
- екран клієнта;
- екран налаштувань.

На рис. 2.4. представлена структурна схема мобільного додатку для сервісу таксі у вигляді деревовидної діаграми, яка показує зв’язки між структурними елементами автоматизованої системи для замовлення таксі.

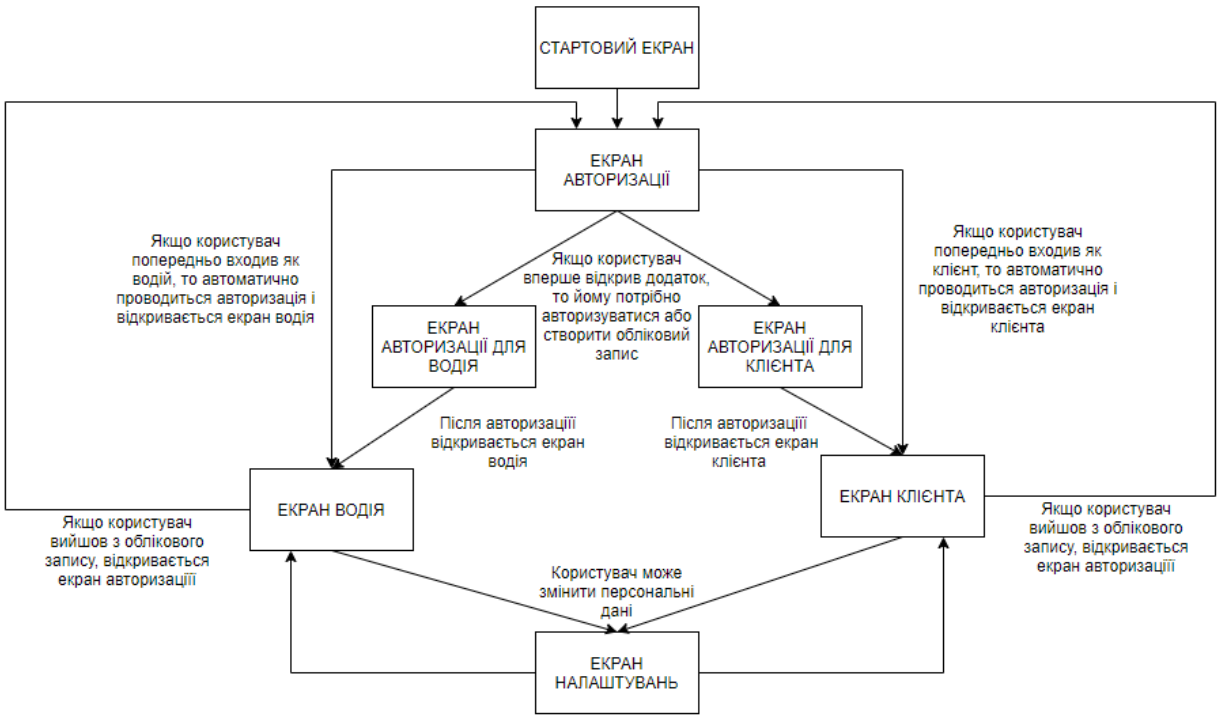


Рис. 2.4. Структурна схема мобільного додатку

Після запуску додатку відкривається *стартовий екран* із зображенням логотипу додатку. Через деякий час здійснюється перехід в *екран авторизації*.

Якщо користувач попередньо входив у додаток як водій, то автоматично проводиться авторизація і відкривається *екран водія*.

Якщо користувач попередньо входив у додаток як клієнт, то автоматично проводиться авторизація і відкривається *екран клієнта*.

Якщо користувач вперше відкрив додаток, то на *екрані авторизації* потрібно обрати бажану кнопку, яка буде вести *на екран авторизації для водія* або *екран авторизації для клієнта*.

На *екрані авторизації для водія* користувач може створити обліковий запис водія та авторизуватись за допомогою нього. Після авторизації відкривається *екран водія*.

На *екрані авторизації для клієнта* користувач може створити обліковий запис клієнта та авторизуватись за допомогою нього. Після авторизації відкривається *екран клієнта*.

На *екрані водія* користувач може вийти з обліково запису, після чого відкриється *екран авторизації*, або перейти до *екрану налаштувань*, в якому можна змінити персональні дані.

На *екрані клієнта* користувач може вийти з обліково запису, після чого відкриється *екран авторизації*, або перейти до *екрану налаштувань*, в якому можна змінити персональні дані.

2.6. Архітектура системи

Система автоматизованого замовлення таксі побудована на клієнт-серверній архітектурі (рис. 2.5.) і складається з двох компонентів:

1) Android–додаток;
2) сервер та база даних Firebase.

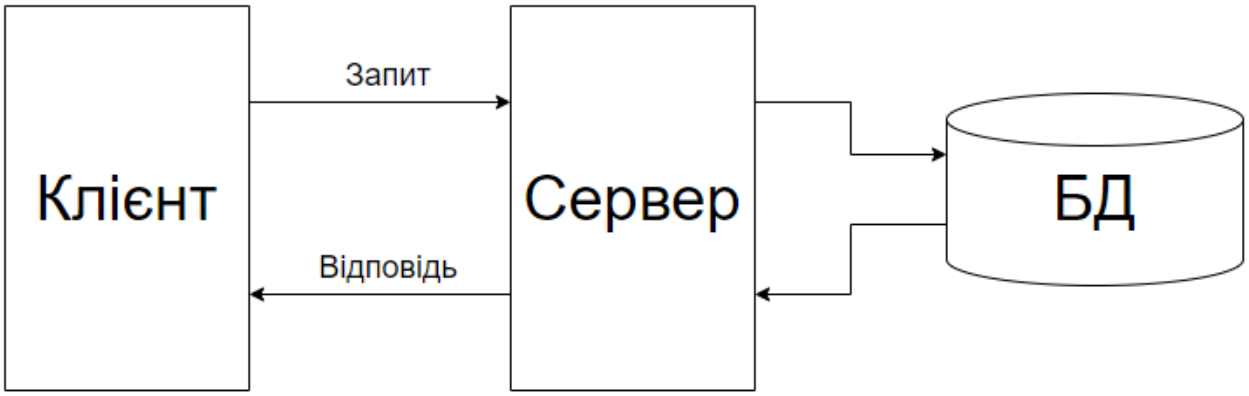


Рис. 2.5. Схема архітектури мобільного додатку

Така архітектура більше всього підходить для даного типу ПО, тому що всі дані зберігаються на сервері і швидко синхронізуються з різними пристроями. Всі обчислення виконуються також на сервері, в результаті чого вимоги до ЕОМ знижуються.

Клієнт передає на сервер дані, які використовуються службою автентифікації та зберігаються в базі даних.

2.7. Схема бази даних системи

Для зберігання та синхронізації даних між клієнтом та сервером була обрана хмарна нереляційна база даних Firebase Realtime Database від компанії Firebase. Вона дозволяє працювати з даними в текстовому форматі JSON та синхронізувати їх в реальному часі з кожним підключеним клієнтом. Дані зберігаються в будь-якому вигляді, не маючи визначеного типу даних.

На відміну від реляційних баз даних, тут не має таблиць та записів. Коли дані додаються в дерево JSON, вони стають вузлом з прив'язаним ключем в існуючій структурі JSON.

Для того, щоб дані могли зчитуватись та записуватись, потрібно встановити наступні правила безпеки:

- «read» – визначає, чи можуть зчитуватись дані з бази даних;
- «write» – визначає, чи можуть дані записуватись в базу даних.

Спроектована база даних має чотири основних вузлів:

- вузол запитів клієнтів «Customers Requests»;
- вузол доступних водіїв «Driver Available»;
- вузол зайнятих водіїв «Driver Working»;
- вузол користувачів «Users».

Вузол «Customers Requests» містить дочірні вузли, ключ яких визначається ключем зареєстрованого користувача, з точним місцезнаходженням клієнтів, які замовили таксі. Структура вузла «Customers Requests» показана на рис. 2.6..

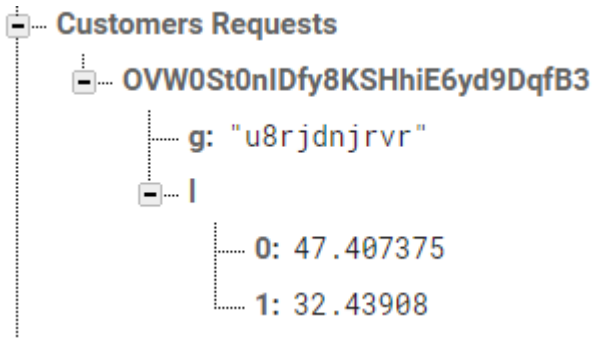


Рис. 2.6. Структура вузла «Customers Requests»

Вузол «Driver Available» містить дочірні вузли, ключ яких визначається ключем зареєстрованого користувача, з точним місцезнаходженням водіїв, які є вільними в даний момент часу та готові прийняти замовлення. Структура вузла «Driver Available» показана на рис. 2.7..

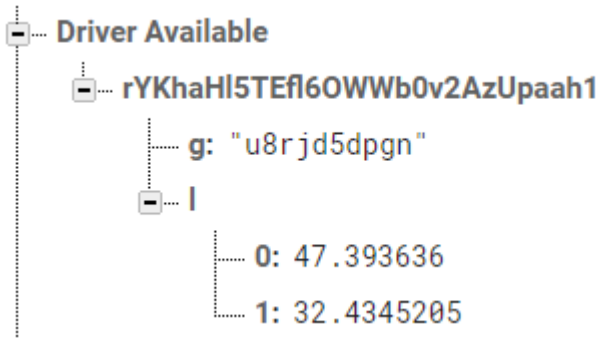


Рис. 2.7. Структура вузла «Driver Available»

Вузол «Driver Working» містить дочірні вузли, ключ яких визначається ключем зареєстрованого користувача, з точним місцезнаходженням водіїв, які вже прийняли замовлення. Структура вузла «Driver Working» показана на рис. 2.8..

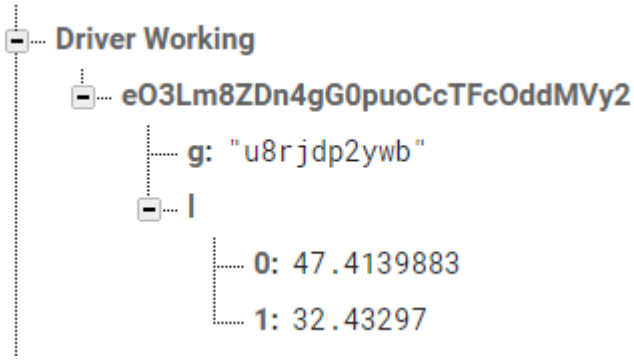


Рис. 2.8. Структура вузла «Driver Working»

Вузол «Users» містить дочірні вузли «Customers» та «Drivers». Вузол «Customers» містить дочірні вузли з персональними даними клієнтів, ключ яких визначається ключем зареєстрованого користувача. Вузол «Drivers» містить дочірні вузли з персональними даними водіїв, ключ яких визначається ключем зареєстрованого користувача. Структура вузла «Users» показана на рис. 2.9..

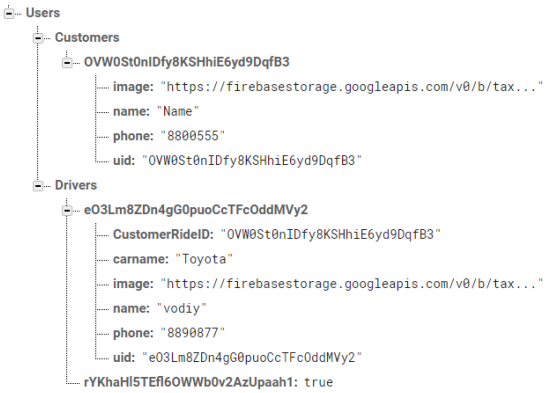


Рис. 2.9. Структура вузла «Users»

Загальний вигляд спроектованої бази даних для мобільного додатку зображений на рис. 2.10..

Рис. 2.10. Спроектована база даних

Висновки до розділу

В ході проектування мобільного додатку було визначено функціональні та нефункціональні вимоги, варіанти використання системи, архітектуру системи побудовано структуру мобільного додатку та схему бази даних системи. Вихідні дані проектування системи можна використовувати для реалізації системи.

РОЗДІЛ 3
РЕАЛІЗАЦІЯ МОБІЛЬНОГО ДОДАТКУ

3.1. Інструменти реалізації

В якості інтегрованого середовища розробки використовувалась Android Studio. Це офіційна IDE для ОС Android, створена компанією Google. Саме тому Android Studio є найкращим вибором для розробки програмного забезпечення під Android-платформу.

В програмі є багато інструментів, які спрощують процес розробки, та вбудований емулятор, який дозволяє перевіряти роботу додатку на пристроях з різними параметрами.

Переваги Android studio перед іншими IDE:

- дозволяє створювати додатки не тільки для смартфонів, але і для інших пристроїв, які працюють на базі операційної системи Android. Наприклад: приставки для телевізорів або розумні годинники;
- тестування коректності роботи додатку проходить в самому емуляторі;
- програма має конструктор елементів інтерфейсу користувача. Це спрощує редагування файлів розмітки;
- дозволяє синхронізувати проект з платформою Firebase без втручання в код.

Для написання програмного коду використовувалась мова програмування високого рівня Java.

3.2. Реалізація серверної часини мобільного додатку

Для роботи з хмарною базою даних та службою автентифікації було проведено інтеграцію мобільного додатку з платформою Firebase за допомогою вбудованого в Android Studio інструменту «Firebase» (рис. 3.1.). Він дозволяє підключити необхідні бібліотеки, не втручаюсь при цьому в код.

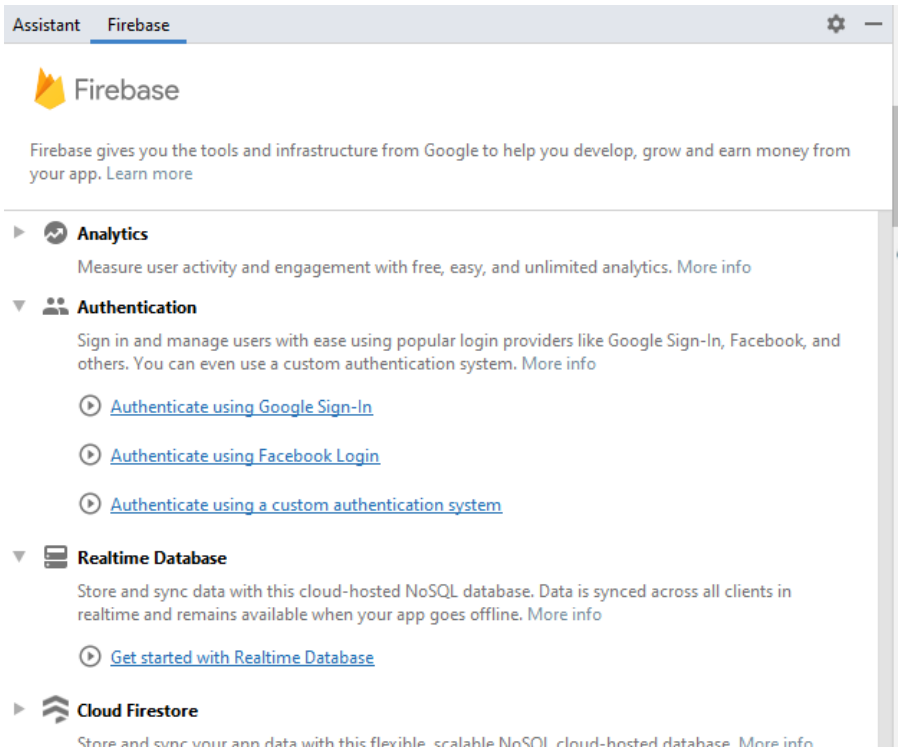


Рис. 3.1. Вигляд панелі «Firebase» в Android Studio

3.3. Реалізація мобільного додатку

Враховуючи особливості програмування для ОС Android, розробку додатку потрібно починати з графічного інтерфейсу, тому спершу були створені елементи інтерфейсу користувача. Вони забезпечують взаємодію користувача з додатком, а їх функціонал описується в програмному коді. Графічний інтерфейс оголошується в XML–файлах.

На рис. 3.2. представлений каталог XML–файлів, які формують розмітку інтерфейсу користувача додатку.

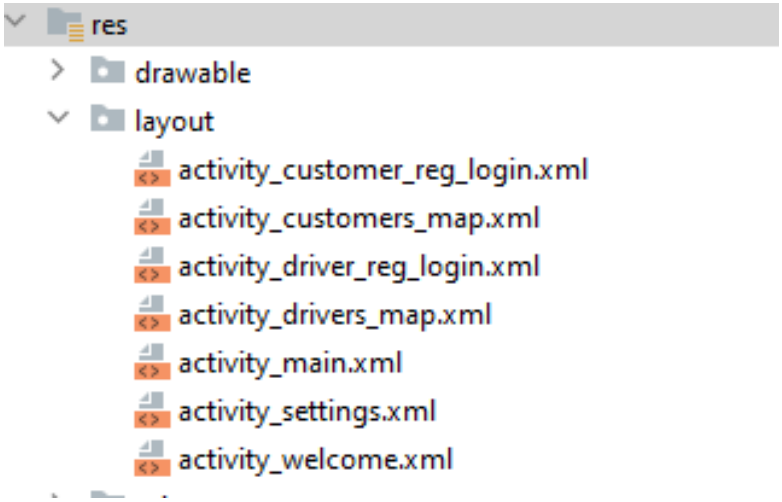


Рис. 3.2. Каталог XML–файлів

Проект включає ряд каталогів, в яких містяться файли програмного коду, що керують поведінкою додатку. Вони підключені до відповідних XML–файлів. На рис. 3.3. представлений каталог файлів Java.

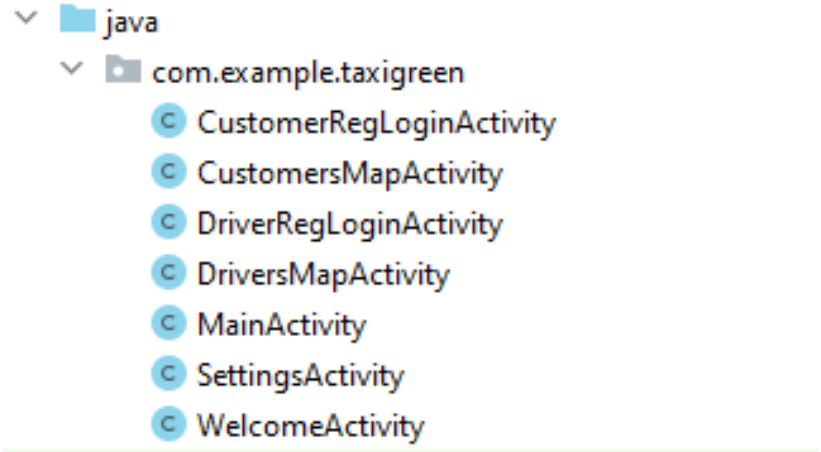


Рис. 3.3. Каталог файлів Java

3.3.1. Стартовий екран

При запуску додатку відкривається стартовий екран із зображенням логотипу додатку (рис. 3.4.) і через 5 секунд здійснюється перехід на екран авторизації.

Рис. 3.4. Стартовий екран

Для цього в програмному коді був створений потік (рис. 3.5.), який робить затримку відображення стартового екрану в п’ять секунд, а потім здійснює перехід на екран авторизації.

```
Thread thread = run() → {
        super.run();
        {
            try
            {
                sleep( millis: 5000);
            }
            catch(Exception e)
            {
                e.printStackTrace();
            }
            finally
            {

                Intent welcomeIntent = new Intent( packageContext: MainActivity.this, WelcomeActivity.class);
                startActivity(welcomeIntent);
            }
        }
};
thread.start();
}
```

Рис. 3.5. Потік «thread»

3.3.2. Екран авторизації

На екрані авторизації присутні кнопки «Я ВОДІЙ» та «Я КЛІЄНТ» (див. рис. 3.6.).

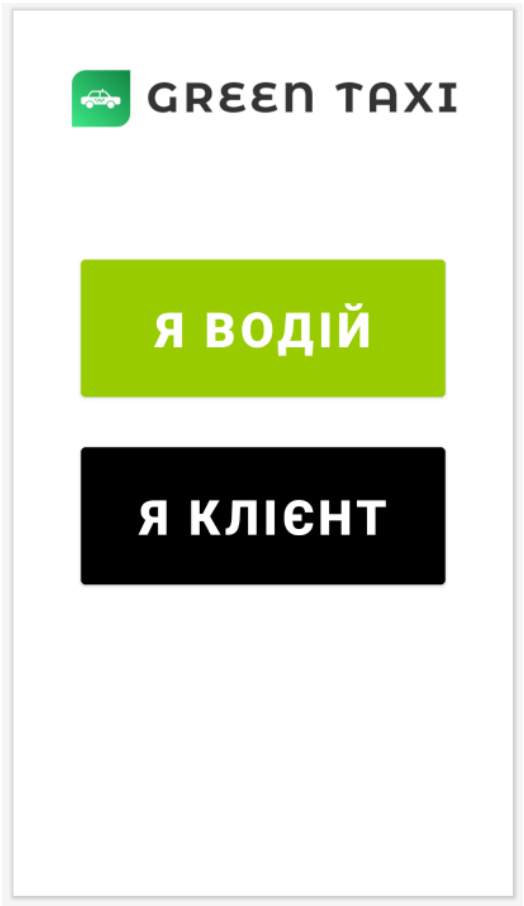


Рис. 3.6. Екран авторизації

При натисканні кнопки «Я ВОДІЙ», проводиться перевірка (рис. 3.7.), чи надав користувач дозвіл на отримання точних координат місцезнаходження пристрою. Якщо дозвіл було надано, то відкривається екран авторизації для водія. Якщо дозвіл не було надано, то з’явиться діалогове вікно з проханням надати дозвіл. Він потрібен для того, щоб додаток зміг визначити точне місцезнаходження пристрою і передати ці дані на сервер.

Рис. 3.7. Перевірка дозволу на отримання місцезнаходження пристрою

Аналогічно проводиться перевірка при натисканні кнопки «Я КЛІЄНТ». Якщо дозвіл було надано, то здійснюється перехід на екран авторизації для клієнта. Якщо дозвіл не було надано, то з’явиться діалогове вікно з відповідним проханням.

Якщо користувач при попередньому запуску додатку проходив авторизацію, то додаток автоматично здійснює вхід в обліковий запис. Для цього проводиться перевірка існування облікового запису в базі даних (рис. 3.8.).

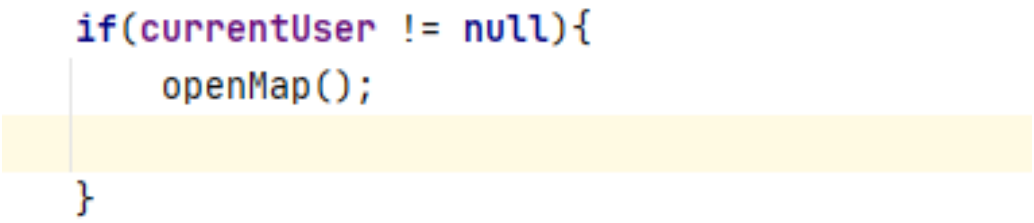


Рис. 3.8. Перевірка існування облікового запису

Якщо обліковий запис існує, то виконується метод «openMap» (рис. 3.9.), який перевіряє, хто авторизувався в додатку (водій чи клієнт) і відкриває відповідний екран.

```
private void openMap() {
    databaseReference.child("Customers").addValueEventListener(new ValueEventListener() {
        @Override
        public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
            if(dataSnapshot.child(mAuth.getCurrentUser().getUid()).exists())
            {
                startActivity(new Intent( packageContext: WelcomeActivity.this, CustomersMapActivity.class));
            } else {
                startActivity(new Intent( packageContext: WelcomeActivity.this, DriversMapActivity.class));
            }

        }

        @Override
        public void onCancelled(@NonNull DatabaseError databaseError) {

        }
    });
}
```

Рис. 3.9. Метод «openMap»

3.3.3. Екран авторизації для водія

На екрані авторизації для водія (рис. 3.10.) присутні два поля з підказками для вводу текстових даних, заголовок «ВХІД» та кнопки: «УВІЙТИ», «НЕ МАЄТЕ ОБЛІКОВО ЗАПИСУ?», «ЗАРЕЄСТРУВАТИСЬ».

Рис. 3.10. Екран авторизації

Щоб зареєструватись в додатку, користувач повинен натиснути кнопку «НЕ МАЄТЕ ОБЛІКОВО ЗАПИСУ?», в результаті чого заголовок зміниться на «РЕЄСТРАЦІЯ» та з’явиться кнопка «ЗАРЕЄСТРУВАТИСЬ», яка була неактивною (рис. 3.11.).

```
question.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        signInBtn.setVisibility(View.INVISIBLE);
        signInBtn.setEnabled(false);
        question.setVisibility(View.INVISIBLE);
        question.setEnabled(false);
        signUpBtn.setVisibility(View.VISIBLE);
        signUpBtn.setEnabled(true);
        driverStatus.setText("РЕЄСТРАЦІЯ");
    }
});
```

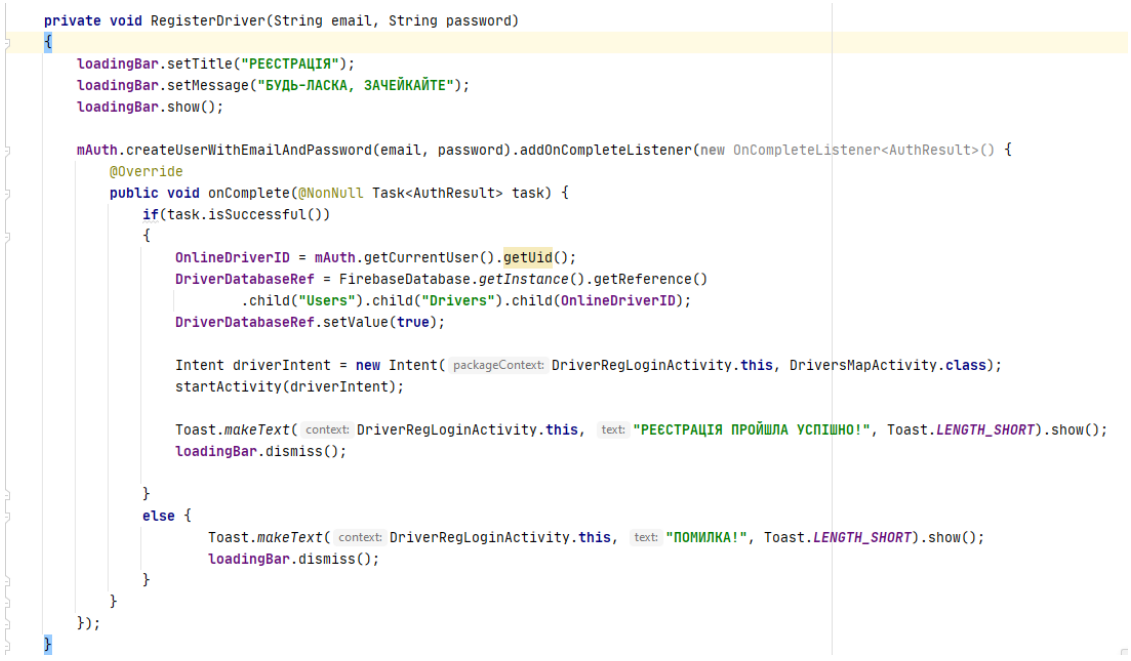
Рис. 3.11. Активація кнопки «ЗАРЕЄСТРУВАТИСЬ» та зміна заголовку

Користувач вводить пошту та пароль у відповідні поля і натискає кнопку «ЗАРЕЄСТРУВАТИСЬ». Обробник подій цієї кнопки (рис. 3.12.) зчитує дані і передає їх в метод «RegisterDriver», який створює новий обліковий запис в базі даних з унікальним ключем та відкриває екран водія (рис. 3.13.).

```
signUpBtn.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        String email = emailET.getText().toString();
        String password = passwordET.getText().toString();

        RegisterDriver(email, password);
    }
});
```

Рис. 3.12. Обробник подій кнопки «ЗАРЕЄСТРУВАТИСЬ» на екрані авторизації водія



```
private void RegisterDriver(String email, String password)
{
    loadingBar.setTitle("РЕЄСТРАЦІЯ");
    loadingBar.setMessage("БУДЬ-ЛАСКА, ЗАЧЕЙКАЙТЕ");
    loadingBar.show();

    mAuth.createUserWithEmailAndPassword(email, password).addOnCompleteListener(new OnCompleteListener<AuthResult>() {
        @Override
        public void onComplete(@NonNull Task<AuthResult> task) {
            if(task.isSuccessful())
            {
                OnlineDriverID = mAuth.getCurrentUser().getUid();
                DriverDatabaseRef = FirebaseDatabase.getInstance().getReference()
                        .child("Users").child("Drivers").child(OnlineDriverID);
                DriverDatabaseRef.setValue(true);

                Intent driverIntent = new Intent( packageContext: DriverRegLoginActivity.this, DriversMapActivity.class);
                startActivity(driverIntent);

                Toast.makeText( context: DriverRegLoginActivity.this,  text: "РЕЄСТРАЦІЯ ПРОЙШЛА УСПІШНО!", Toast.LENGTH_SHORT).show();
                loadingBar.dismiss();

            }
            else {
                    Toast.makeText( context: DriverRegLoginActivity.this,  text: "ПОМИЛКА!", Toast.LENGTH_SHORT).show();
                    loadingBar.dismiss();
            }
        }
    });
}
```

Рис. 3.13. Метод «RegisterDriver»

Якщо користувач має обліковий запис водія, то він вводить пошту та пароль у відповідні поля і натискає кнопку «УВІЙТИ». Обробник подій цієї кнопки (рис. 3.14.) зчитує дані і передає їх в метод «SignInDriver» (рис. 3.15.), який проводить авторизацію користувача в системі і відкриває екран водія. В іншому випадку зв’явится повідомлення про помилку.

```
signInBtn.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        String email = emailET.getText().toString();
        String password = passwordET.getText().toString();

        SignInDriver(email, password);

    }
});
```

Рис. 3.14. Обробник подій для кнопки «УВІЙТИ» на екрані водія

```
private void SignInDriver(String email, String password)
{
    loadingBar.setTitle("ВХІД");
    loadingBar.setMessage("БУДЬ-ЛАСКА, ЗАЧЕЙКАЙТЕ");
    loadingBar.show();

    mAuth.signInWithEmailAndPassword(email, password).addOnCompleteListener(new OnCompleteListener<AuthResult>() {
        @Override
        public void onComplete(@NonNull Task<AuthResult> task) {
            if(task.isSuccessful())
            {
                Toast.makeText( context: DriverRegLoginActivity.this,  text: "УСПІШНИЙ ВХІД!", Toast.LENGTH_SHORT).show();
                loadingBar.dismiss();
                Intent driverIntent = new Intent( packageContext: DriverRegLoginActivity.this, DriversMapActivity.class);
                startActivity(driverIntent);
            }
            else {
                Toast.makeText( context: DriverRegLoginActivity.this,  text: "ПОМИЛКА! СПРОБУЙТЕ ЩЕ РАЗ", Toast.LENGTH_SHORT).show();
                loadingBar.dismiss();
            }
        }
    });
}
```

Рис. 3.15. Метод «SignInDriver»

3.3.4. Екран авторизації для клієнта

Екран авторизації для клієнта має ідентичний набір елементів як і екран авторизації для водія (рис. 3.16.). Різниця полягає лише в тому, що користувач реєструється або авторизується в якості клієнта. Після створення або авторизації облікового запису відкривається екран клієнта.

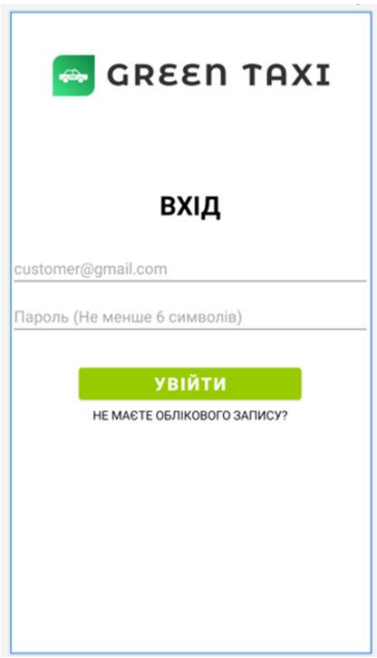


Рис. 3.16. Екран авторизації для клієнта

3.3.5. Екран водія

При переході на екран водія (рис. 3.17.) завантажується карта, на якій відображається маркер з місцезнаходженням користувача.

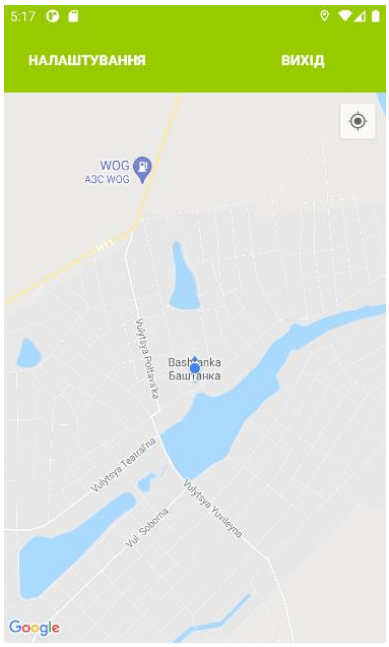


Рис. 3.17. Екран водія

Для цього у файлі розмітки інтерфейсу екрану водія був створений фрагмент, який відображає карту Google Maps (рис. 3.18.), а в програмному коді було проведено ініціалізацію цього фрагменту та створено метод «onMapReady» (рис. 3.19.), який перевіряє чи було надано дозвіл на отримання точного місцезнаходження мобільного пристрою. Якщо дозвіл був наданий, то на карті з’явиться маркер користувача.

```
<fragment
    android:id="@+id/map"
    android:name="com.google.android.gms.maps.SupportMapFragment"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:layout_marginTop="70dp"

    />
```

Рис. 3.18. Фрагмент карти Google Maps у файлі розмітки екрану водія

```
@Override
public void onMapReady(GoogleMap googleMap) {
    mMap = googleMap;

    buildGoogleApiClient();
    if (ActivityCompat.checkSelfPermission( context: this, Manifest.permission.ACCESS_FINE_LOCATION) !=
            PackageManager.PERMISSION_GRANTED && ActivityCompat.checkSelfPermission( context: this, Manifest.permission.ACCESS_COARSE_LOCATION) !=
            PackageManager.PERMISSION_GRANTED) {
        return;
    }
    mMap.setMyLocationEnabled(true);
}
```

Рис. 3.19. Метод «onMapReady»

Коли користувач знаходиться на екрані водія, то його ключ в базі даних заноситься у вузол вільних водіїв, але коли він отримує замовлення, то ключ видаляється з вузла вільних водіїв і заноситься у вузол зайнятих водіїв (рис. 3.20.).

```
String userID = FirebaseAuth.getInstance().getCurrentUser().getUid();
DatabaseReference DriverAvalablityRef = FirebaseDatabase.getInstance().getReference().child("Driver Available");
GeoFire geoFireAvailablity = new GeoFire(DriverAvalablityRef);


DatabaseReference DriverWorkingRef = FirebaseDatabase.getInstance().getReference().child("Driver Working");
GeoFire geoFireWorking = new GeoFire(DriverWorkingRef);


switch (customerID)
{
    case "":
        geoFireWorking.removeLocation(userID);
        geoFireAvailablity.setLocation(userID, new GeoLocation(location.getLatitude(), location.getLongitude()));
        break;
    default:
        geoFireAvailablity.removeLocation(userID);
        geoFireWorking.setLocation(userID, new GeoLocation(location.getLatitude(), location.getLongitude()));
        break;
}
```

Рис. 3.20. Якщо водій отримує замовлення, то він стає недоступним

Після того як водій отримав замовлення, на його екрані зв’явиться маркер клієнта та блок з інформацією про нього (рис. 3.21.). Якщо водій захоче зв’язатися з клієнтом, то він може натиснути на значок телефонної трубки, щоб зателефонувати йому. Якщо замовлення буде скасовано, то маркер та блок інформації зникнуть з екрану водія.

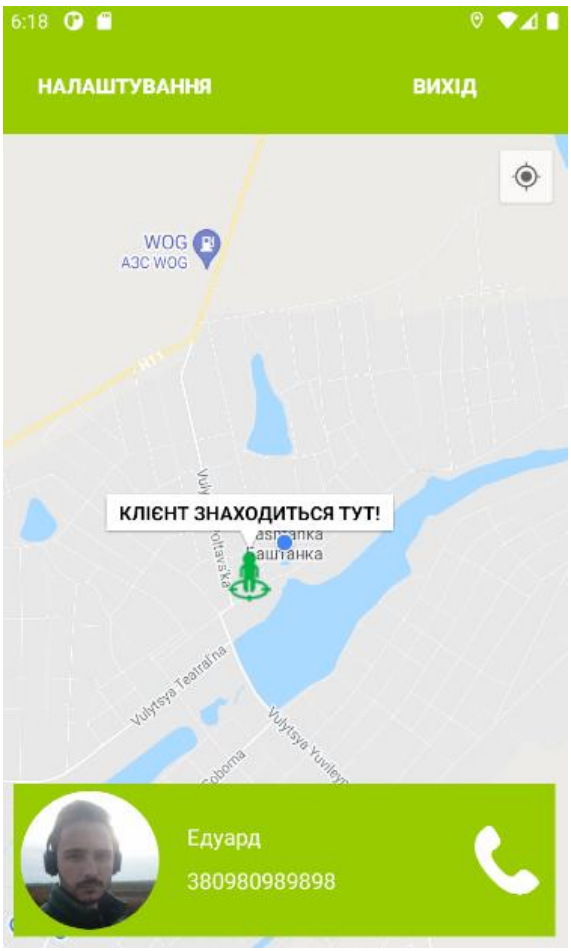


Рис. 3.21. Поява маркера клієнта та блоку інформації про нього

На екрані водія присутні кнопки «НАЛАШТУВАННЯ» та «ВИХІД». При натисканні першої кнопки здійснюється перехід на екран налаштувань, а при натисканні другої – вихід з облікового запису та перехід на екран авторизації.

3.3.6. Екран клієнта

На екран клієнта (рис. 3.22.) аналогічно завантажується карта, на якій відображається маркер з місцезнаходженням користувача і також присутні кнопки «НАЛАШТУВАННЯ» та «ВИХІД», які виконують ті самі функції. Але окрім них на екрані розташована кнопка «ЗАМОВИТИ ТАКСІ», при натисканні якої почнеться пошук вільного водія поблизу (рис. 3.23.)

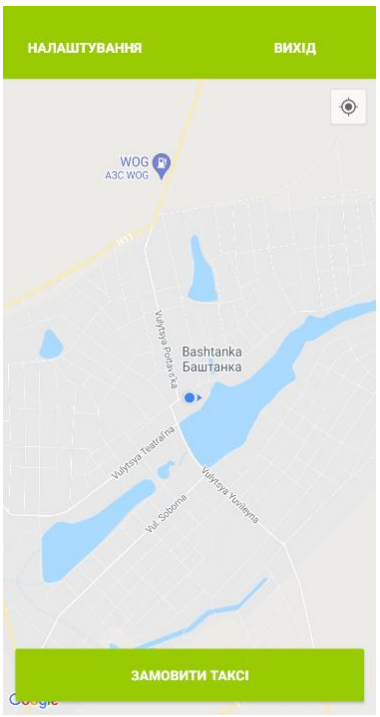


Рис. 3.22. Екран клієнта

```
private void getNearbyDrivers()
{
    GeoFire geoFire = new GeoFire(DriversAvailableRef);
    GeoQuery geoQuery = geoFire.queryAtLocation(new GeoLocation(CustomerPosition.latitude, CustomerPosition.longitude), radius);
    geoQuery.removeAllListeners();

    geoQuery.addGeoQueryEventListener(new GeoQueryEventListener() {
        @Override
        public void onKeyEntered(String key, GeoLocation location) {
            if(!driverFound && requestType)
            {
                driverFound = true;
                driverFoundID = key;

                DriversRef = FirebaseDatabase.getInstance().getReference().child("Users").child("Drivers").child(driverFoundID);
                HashMap driverMap = new HashMap();
                driverMap.put("CustomerRideID", customerID);
                DriversRef.updateChildren(driverMap);

                GetDriverLocation();
            }
        }
```

Рис. 3.23. Пошук вільного водія в базі даних

Якщо від серверу прийде відповідь про те, що є доступні водії, то клієнт робить запит на отримання даних про місцезнаходження водія з бази даних. Після отримання координат на екрані з’являється маркер водія та блок інформації про нього (рис. 3.24.), на якому присутня кнопка у вигляді телефонної трубки, яка

дозволяє зв’язатися з клієнтом. На кнопці «НАТИСНІСТЬ, ЩОБ ВІДМОВИТИСЬ» з’явиться інформація про відстань до водія.

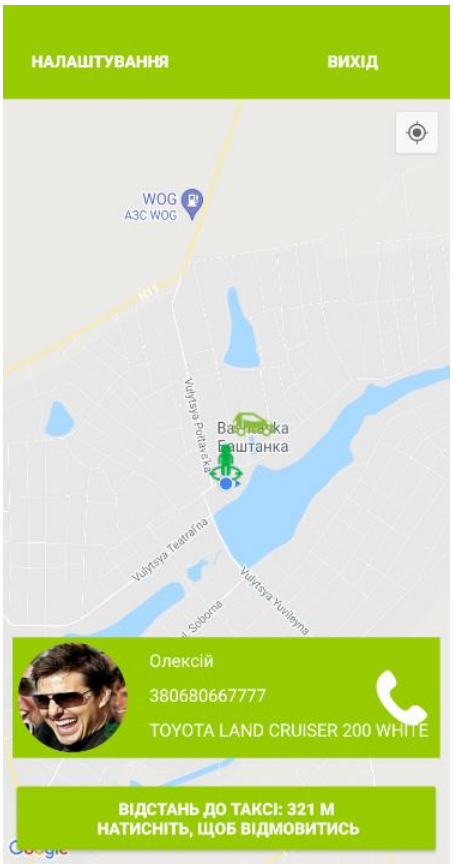


Рис. 3.24. Поява маркера водія та блоку інформації про нього

Якщо клієнт захоче скасувати замовлення, то йому необхідно натиснути кнопку «НАТИСНІСТЬ, ЩОБ ВІДМОВИТИСЬ», після чого маркер водія та інформація про нього зникнуть з екрану. Водій стане знову доступним для інших замовлень.

3.3.7. Екран налаштувань

На екрані налаштувань присутнє фото користувача, кнопки для збереження та скасування змін, кнопка «ЗМІНИТИ ФОТО» і два поля для вводу імені та номеру телефону користувача (рис. 3.25.). Вся введена інформація завантажується в базу даних (рис. 3.26.).

Рис. 3.25. Екран налаштувань для клієнта

```
private void ValidateAndSaveOnlyInformation() {

    if(TextUtils.isEmpty(nameET.getText().toString())){
        Toast.makeText( context: this,  text: "ВВЕДІТЬ ІМ'Я", Toast.LENGTH_SHORT).show();
    }
    else if(TextUtils.isEmpty(phoneET.getText().toString())){
        Toast.makeText( context: this,  text: "ВВЕДІТЬ НОМЕР", Toast.LENGTH_SHORT).show();
    }
    else if(getType.equals("Drivers") && TextUtils.isEmpty(carET.getText().toString())){
        Toast.makeText( context: this,  text: "ВВЕДІТЬ МАРКУ АВТОМОБІЛЯ", Toast.LENGTH_SHORT).show();
    }
    else {
        HashMap<String, Object> userMap = new HashMap<>();
        userMap.put("uid",mAuth.getCurrentUser().getUid());
        userMap.put("name",nameET.getText().toString());
        userMap.put("phone",phoneET.getText().toString());

        if(getType.equals("Drivers"))
        {
            userMap.put("carname",carET.getText().toString());
        }

        databaseReference.child(mAuth.getCurrentUser().getUid()).updateChildren(userMap);


        if(getType.equals("Drivers"))
        {
            startActivity(new Intent( packageContext: SettingsActivity.this, DriversMapActivity.class));
        }else{
            startActivity(new Intent( packageContext: SettingsActivity.this, CustomersMapActivity.class));
        }
    }
}
```

Рис. 3.26. Завантаження інформації про користувача в базу даних

Якщо екран налаштувань буде відкритий з екрану водія, то на екрані налаштувань буде присутнє ще одне поле, в якому водій повинен вказати марку автомобіля (рис. 3.27.).

Рис. 3.27. Екран налаштувань для водія

Висновки до розділу

В ході розробки програмного забезпечення були визначенні інструменти реалізації. В якості середовища розробки використовувалась програма Android Studio від компанії Google. Програмний код був написаний на мові програмування високо рівня Java.

При реалізації клієнтської частини були створені сім екранів мобільного додатку, які задовольняють всім визначеним на етапі проектування вимогам: стартовий екран, екран авторизації, екран авторизації для водія, екран авторизації для клієнта, екран водія, екран клієнта, екран налаштувань.

ВИСНОВКИ

В процесі виконання дипломного проекту були розглянуті методи розробки мобільних додатків для сервісу таксі під платформу Android, види додатків для замовлення таксі, їх класифікація та особливості. Зібрані дані використовувались для проектування та реалізації мобільного додатку для замовлення таксі під операційну систему Android. Додаток поєднує лаконічний інтерфейс та простоту використання, дозволяє швидко та зручно викликати таксі.

При розробці програмного забезпечення були вирішені наступні задачі:

1) вивчені відомі методи розробки мобільних додатків для виклику таксі;
2) проведений огляд існуючих застосунків для виклику таксі;
3) визначено вимоги та спроектовано мобільний додаток;
4) реалізовано та проведення тестування додатку.

Розроблене програмне забезпечення задовольняє наступним вимогам:

1) Мобільний додаток один для водіїв та клієнтів;
2) Користувач має можливість реєстрації та авторизації за допомогою електронної пошти і паролю;
3) Користувач може змінювати свої персональні дані;
4) Клієнт може замовити таксі в один клік;
5) Мобільний додаток демонструє переміщення водія та клієнта на карті;
6) Клієнт може проглянути інформацію про водія;
7) Водій може проглянути інформацію про клієнта;
8) Водій та клієнт можуть зв’язатися один з одним через додаток;
9) Клієнт може скасувати замовлення;
10) Користувач має можливість зміни облікового запису.

Таким чином, всі поставлені в проекті задачі були вирішенні, мета проекту була досягнута. Практична цінність дипломного проекту полягає в тому, що розроблена система може використовуватись для спрощення роботи компаній, які займаються таксомоторними перевезеннями. Автоматизація процесів дозволить зменшити витрати та підвищити ефективність обслуговування клієнтів.